Otto Zublasing

Monte Carlo Simulation für ein gekoppeltes Round Robin System

Analyse des Systems und Simulationsergebnisse / Modulstruktur und Quelltexte des Pascal Programms

Diplomarbeit

Diplom.de

Bibliografische Information der Deutschen Nationalbibliothek:

Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

ISBN: 9783832407469

Otto Zublasing

Monte Carlo Simulation für ein gekoppeltes Round Robin System

Analyse des Systems und Simulationsergebnisse / Modulstruktur und Quelltexte des Pascal Programms

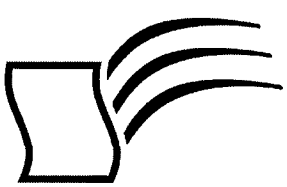
Diplom.de

Otto Zublasing

Monte Carlo Simulation für ein gekoppeltes Round Robin System

Analyse des Systems und Simulationsergebnisse / Modulstruktur und Quelltexte des Pascal Programms

Diplomarbeit
an der Georg-Simon-Ohm-Fachhochschule Nürnberg
Juli 1995 Abgabe



***Diplomarbeiten* Agentur**
Dipl. Kfm. Dipl. Hdl. Björn Bedey
Dipl. Wi.-Ing. Martin Haschke
und Guido Meyer GbR

Hermannstal 119 k
22119 Hamburg

agentur@diplom.de
www.diplom.de

ID 746
Zublasing, Otto: Monte Carlo Simulation für ein gekoppeltes Round Robin System: Analyse des Systems und Simulationsergebnisse / Modulstruktur und Quelltexte des Pascal Programms / Otto Zublasing - Hamburg: Diplomarbeiten Agentur, 1998
Zugl.: Nürnberg, Fachhochschule, Diplom, 1995

Dipl. Kfm. Dipl. Hdl. Björn Bedey, Dipl. Wi.-Ing. Martin Haschke & Guido Meyer GbR
Diplomarbeiten Agentur, http://www.diplom.de, Hamburg
Printed in Germany

Vorwort

Die Diplomarbeit gliedert sich in zwei Teile, die getrennt in zwei Bände ausgegeben werden:

- Teil I der Diplomarbeit: Analyse des Systems und Simulationsergebnisse
- Teil II der Diplomarbeit: Modulstruktur und Quelltexte des PASCAL Programms

Im vorliegenden ersten Teil werden die Systemkomponenten, sowie ihre Funktionsweise besprochen. Außerdem erfolgt eine Beschreibung der Realisierung des Simulationsprogramms, und eine Überprüfung der implementierten Algorithmen. Abschließend werden einige Simulationen durchgeführt und besprochen.

Der zweite Teil der Diplomarbeit enthält eine Übersicht über die einzelnen Programmoduln und den darin implementierten Funktionen. Der Ablauf der wichtigsten Algorithmen des Simulationsprogramms wird an Hand von Struktogrammen aufgezeigt. Im Anschluß befinden sich die Quelltexte der PASCAL Moduln[1].

Bei dem vorliegenden Text handelt es sich um eine überarbeitete Fassung der Diplomarbeit. Im wesentlichen wurde der Originaltext (der noch auf einem Atari Mega-ST geschrieben wurde, der mich treu durch mein Studium begleitete) unverändert übernommen und nach Microsoft Word für Windows© konvertiert. An einigen Stellen konnte ich allerdings nicht umhin, nachträglich doch einige Anmerkungen anzubringen, die allerdings als solche kenntlich gemacht wurden.

[1] Wie im Vorwort von Teil II der Diplomarbeit beschrieben, werden in dieser überarbeiteten Fassung auch die Turbo C Quellen des Simulationsprogramms ausgegeben.

Inhalt

1 Einleitung

An dieser Stelle erfolgt ein kleiner Überblick über die Themenkreise, die im Laufe des Textes angesprochen werden. Fachbegriffe und Abkürzungen, die in diesem oder in einem der folgenden Abschnitte verwendet werden, können in „Anhang A" nachgeschlagen werden, sofern sie nicht an jeweiliger Stelle erklärt sind.

Die vorliegende Arbeit befaßt sich mit der Analyse eines gekoppelten Round Robin Systems mit Hilfe der Monte Carlo Methode. Der Begriff Monte Carlo Simulation wurde in den vierziger Jahren von Nicholas Metropolis und Stanislaw Ulam geprägt[2]. Die Monte Carlo Simulation wird verwendet für die Analyse von statistischen Problemen mit bekannten Wahrscheinlichkeitsverteilungen. Ein statistisches Problem liegt beispielsweise bei der Simulation von Warteschlangenmodellen oder Bedienungssystemen vor. Dort gilt, daß die Zwischenankunftszeit, sowie die Bedienungszeit einer Anforderung unabhängig von den Zwischenankunftszeiten oder Bedienungszeiten aller vorhergehenden Anforderungen ist. Die Zwischenankunftszeit einer Anforderung ist dabei die Zeit, die zwischen der Ankunft zweier aufeinanderfolgender Anforderungen vergeht. Das heißt, die Zwischenankunfts- und die Bedienungszeit einer Anforderung ist unabhängig von dem Zeitpunkt, oder der Reihenfolge, an dem die Anforderung das System betritt, und für beide Zeiten gibt es charakteristische Verteilungsfunktionen.

Bei der Simulation eines Bedienungssystems, das eine Folge von Anforderungen verarbeiten soll, bedeutet das, daß das System nicht für jeden beliebigen Zeitpunkt untersucht werden muß. Dies würde zu einer sehr großen Anzahl von Berechnungen führen, die in endlicher Zeit nicht mehr durchführbar wären. Bei der Monte Carlo Simulation reicht es vielmehr aus, die Simulation an Hand einer Auswahl von Anforderungen durchzuführen, deren Ankunftszeiten zufällig verteilt sind, und wenn der Betrachtungszeitraum der gesamten Untersuchung „erheblich größer" ist als die Zwischenankunftszeit der Anforderungen, dann erhält man ein Simulationsmodell, dessen Verhalten ziemlich gut mit der Praxis übereinstimmt.

Bei einem Round Robin System handelt es sich um ein sogenanntes Zeitscheibenverfahren, das auch Umlaufverfahren genannt wird. Das heißt, mehrere lauffähige Prozesse gleicher Priorität dürfen nur in einem bestimmten Zeitintervall, der Zeitscheibe, rechnen und müssen anschließend den Prozessor einem anderen Prozeß überlassen (siehe Abbildung 2-1 im nächsten Abschnitt). In diesem Kontext sind dabei die Begriffe „Prozeß" und „Anforderung" gleichbedeutend.

Wenn ein Prozeß das System betritt, wird er zunächst an das Ende der Warteschlange, der sogenannten Bereitliste, eingetragen. Dort muß jeder Prozeß solange warten, bis ein Prozessorzugriff möglich ist und er für die Dauer eine Zeitscheibe rechnen darf. Beim Round

[2] Aus dem Artikel „Wege aus der Unberechenbarkeit" von Joseph Traub und Henryk Wozniakowski. Spektrum der Wissenschaft, April 1994.

Robin Verfahren sind dabei die Zeitscheibenlängen konstant. Nach Ablauf des Zeitquantums wird der Prozeß[3] wieder an das Ende der Warteschlange angehängt, und der an erster Stelle stehende Prozeß der Warteschlange wird nun vom Prozessor verarbeitet; jeder Prozeß wird also wieder mit seiner Restbedienzeit in die Warteschlange eingereiht, durchläuft das System mehrfach, bis er endgültig bearbeitet ist.

Die Wahl der Zeitscheibenlänge kann dabei entscheidend sein für die Anzahl der endgültig verarbeiteten Prozesse innerhalb des Betrachtungszeitraums, was als Durchsatz oder die Performance des Systems bezeichnet wird. Wenn die Zeitscheibe zu klein gewählt wird, ist der Zeitaufwand für die Prozeßwechsel unverhältnismäßig groß. Geht dagegen die Zeitscheibe gegen unendlich (ist also das Zeitintervall sehr viel größer als die mittlere Bearbeitungszeit der Prozesse), verhält sich das System nach dem FCFS Verfahren (First Come First Served). Bei diesem Verfahren darf jeder Prozeß in der Reihenfolge seines Eintreffens, nachdem ihm der Prozessor zugeteilt wurde, solange rechnen, bis er abgearbeitet ist.

Das Round Robin Verfahren begünstigt kurze und benachteiligt lange Prozesse. Dies ist leicht einzusehen, wenn man sich vorstellt, daß die Rechenzeit eines Prozesses nicht größer ist als die Zeitscheibe[4], was dazu führt, daß kurze Prozesse innerhalb eines Zeitquantums abgearbeitet werden können und nicht von neuem in die Warteschlange eingereiht werden müssen. Lange Prozesse, deren Rechenzeit größer als eine Zeitscheibe ist[5], müssen dagegen öfters warten und befinden sich wiederholt in der Warteschlange, bis sie endgültig abgearbeitet sind.

Bei diesem Verfahren ist deswegen darauf zu achten, daß die Prozesse nicht zu lang sind, da sie sonst „ausgehungert" werden. Das heißt, daß Prozesse mit sehr langen Rechenzeiten so oft verdrängt werden, daß es unverhältnismäßig lange dauert, bis sie abgearbeitet sind. Das ist ein Problem der langfristigen Ablaufplanung, in der unter anderem entschieden wird, wie groß die Zeitscheiben gewählt werden sollen, um eine möglichst günstige Performance zu erhalten. Es kann deshalb durchaus von Nutzen sein, die Zeitscheiben während des Betriebsablaufs eines so organisierten Rechners den momentanen Anforderungen des Rechenbetriebs anzupassen, um „Stoßzeiten" zu entlasten oder „Leerzeiten" besser auszulasten.

Im Gegensatz zum Round Robin Verfahren, begünstigt das FCFS Verfahren lange Prozesse und benachteiligt kurze. Wie sich bereits vermuten läßt, liegt das daran, daß ein Prozeß so lange rechnen darf, bis er abgearbeitet ist. Die Verweilzeit für kurze Prozesse steigt allerdings sehr stark an, da sie eventuell auf den Prozessorzugriff sehr lange warten müssen, wenn der Prozessor durch Prozesse mit langer Rechenzeit blockiert wird. Der Vorteil dieses Verfahrens liegt in der Einsparung der Prozeßwechselzeiten, da die einzelnen Prozesse den Prozessor nur noch einmal betreten und einmal verlassen, im Gegensatz zu dem zeitaufwendigen zyklischen Wechsel der einzelnen Prozesse zwischen Warteschlange und Prozessor beim Zeitscheiben-verfahren.

[3] Anmerkung zur überarbeiteten Fassung: Das heißt, wenn der Prozeß noch nicht vollständig abgearbeitet ist.
[4] Anmerkung zur überarbeiteten Fassung: Beziehungsweise wenn bei gegebener Zeitscheibenlänge die Prozesse deren Rechenzeit kleiner der Zeitscheibenlänge sind häufiger auftreten.
[5] Anmerkung zur überarbeiteten Fassung: Beziehungsweise wenn bei gegebener Zeitscheibenlänge die Prozesse deren Rechenzeit größer der Zeitscheibenlänge sind häufiger auftreten.

2 Problemstellung

Bei der Aufgabenstellung handelt es sich um zwei Systeme, die auf ein gemeinsames Betriebsmittel zugreifen. Man kann sich das im wesentlichen als ein Rechensystem mit zwei parallel angeordneten Prozessoren vorstellen, die eine Menge von Anforderungen oder Prozesse verarbeiten, die der Reihe nach aus je einer Warteschlange entnommen werden. Die Anforderungen können nun einen Zugriff auf das gemeinsame Betriebsmittel, zum Beispiel auf die Festplatte oder den Drucker, anfordern. Da ein gleichzeitiger Zugriff auf das Betriebsmittel nicht möglich ist, und um dies in einem realen Betriebssystem weitgehend auszuschließen, muß jede Anforderung vor Verarbeitung durch den Prozessor diese Absicht des Betriebsmittelzugriffs ankündigen. Wenn nun an beiden Prozessoren Anforderungen zur Bearbeitung anstehen, die gleichzeitig auf das Betriebsmittel zugreifen wollen, muß eine der beiden warten.

Zusammenfassend bedeutet das, daß der Durchsatz beider Systeme abhängig von der gemeinsamen Verwendung des Betriebsmittels ist, da eine Anforderung innerhalb eines Systems durch eine Anforderung im anderen System blockiert werden kann. Es handelt sich also um ein gekoppeltes System, wie es in Abbildung 2-1 zu sehen ist.

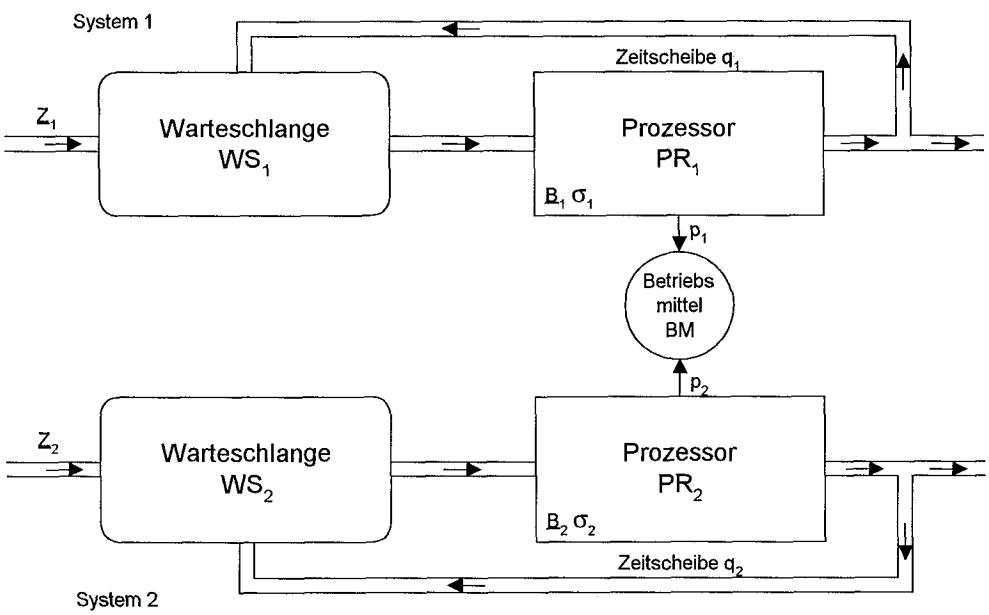


Abbildung 2-1: Die Komponenten des Systems und ihre Parameter.

2.1 Systembeschreibung

Aus welchen Komponenten besteht nun das System, und wie ist deren Arbeitsweise? Wie die Abbildung 2-1 zeigt, sind beide Teilsysteme gleich aufgebaut. Die Bezeichnungen, die während des gesamten Textes und in der Implementierung verwendet werden, unterscheiden sich nur durch eine Ziffer, die die Zugehörigkeit zum System 1 oder zum System 2 andeutet.

Zunächst treffen nacheinander Anforderungen ein, die der Reihe nach an die Warteschlange WS_1 oder WS_2 angefügt werden. Der Erwartungswert der Zwischenankunftszeit eines Ankunftsstroms $\underline{Z}_1$ oder $\underline{Z}_2$ ist exponentialverteilt, was den Beobachtungen aus der Praxis für die Ankunftszeit zweier aufeinanderfolgender Kunden eines Bedienungssystems entspricht (Vergleiche „Anhang B Exponentialverteilung"). Die Bedienungszeiten der Anforderungen können sowohl exponentialverteilt, mit dem Erwartungswert $\underline{B}_1$ beziehungsweise $\underline{B}_2$, als auch normalverteilt sein, mit der zusätzlichen Angabe der Standardabweichung σ_1 oder σ_2. Im System werden prinzipiell zwischen drei Arten von Ankunftsströmen unterschieden:

- Ein gemeinsamer Ankunftsstrom mit zufälliger Aufteilung auf beide Teilsysteme
- Ein gemeinsamer Ankunftsstrom mit bestimmter Aufteilung auf beide Teilsysteme
- Zwei getrennte Ankunftsströme

Wenn im Prozessor PR_1 oder PR_2 keine Anforderung bedient wird, wird die erste Anforderung aus der Warteschlange entnommen und dem Prozessor zugewiesen. Die beiden Prozessoren sind dabei nicht gleich „getaktet", das heißt, sie arbeiten zeitlich unabhängig voneinander. Da das System nach dem Round Robin Verfahren arbeitet, wird jede Anforderung dem Prozessor nur für ein fest vorgegebenes Zeitintervall q_1 oder q_2 zugeteilt. Ist die Bedienungszeit einer Anforderung größer als diese Zeitscheibe, wird die Anforderung nach Ablauf des Zeitquantums verdrängt, und wieder an die Warteschlange angehängt. Dies geschieht solange, bis die Anforderung endgültig abgearbeitet ist. Wenn die Restverarbeitungszeit der Anforderung kleiner ist als die Zeitscheibenlänge, verläßt die Anforderung vor Ablauf der Zeitscheibe den Prozessor. Ist die Rechenzeit einer Anforderung vollständig aufgebraucht, verläßt sie das System, und wird nicht wieder in die Warteschlange eingereiht.

Jede Anforderung in beiden Systemen kann mit der Wahrscheinlichkeit p_1 oder p_2 auf das gemeinsame Betriebsmittel BM zugreifen. Die Entscheidung, ob eine Anforderung auf das Betriebsmittel zugreift, wird dabei einmalig vor dem Betreten des Systems festgelegt, und bleibt für jede Zeitscheibe für den gesamten Bedienungszeitraum der Anforderung erhalten.

Um ein Aushungern zu vermeiden, wird eine Möglichkeit angeboten, Anforderungen eine größere Zeitscheibe zuzuweisen. Das heißt, überschreitet eine Anforderung wegen mehrmaliger und langfristiger Aufenthalte in der Warteschlange eine vorgegebene maximale Verweilzeit, darf sie beim nächsten Prozessorzugriff länger rechnen, als die Zeitscheiben q_1 oder q_2 es ursprünglich vorsahen. Dies gilt insbesondere auch für alle zukünftigen Zeitscheiben einer Anforderung, bis sie das System verläßt, da ihre Verweilzeit ständig über der maximalen Verweilzeit liegt.

Das beschriebene System entspricht in dieser Form weitgehend dem eines Universalrechenautomaten nach von Neumann. Der Prozessor besteht aus dem Leit- und Rechenwerk für die Steuerung und Durchführung der Berechnungen. Die Warteschlange ist das Speicherwerk, in dem sich die zu verarbeitenden Prozesse befinden. Der Zu- und Abgang der Anforderungen in das beziehungsweise aus dem System erfolgt über das Ein- und Ausgabewerk. Bei dem Betriebsmittel könnte es sich beispielsweise um einen Drucker, ein Diskettenlaufwerk, eine Festplatte oder ähnliches handeln.

Das Simulationsmodell hat allerdings durchaus auch einen allgemeinen Charakter. Man kann beispielsweise einen Prozeß auch als Kunde, und den Prozessor als Bedienungsstation ansehen. Die Systemkomponenten können somit auch Teile eines Bedienungssystems, wie zum Beispiel eine Tankstelle oder ein Kaufhaus sein. Das Betriebsmittel wäre dann eine gemeinsame Kasse oder eine Zapfsäule, auf die zwei Verkäufer nicht gleichzeitig zugreifen können. Es sind natürlich auch andere Systeme denkbar, die ähnlich aufgebaut sind, und die mit dem vorliegenden Modell untersucht werden können.

2.2 Prioritätssteuerung

Bei der Vergabe der Prozessoren an die Anforderungen mit Betriebsmittelzugriff soll keine Verdrängung stattfinden. Das heißt, wenn innerhalb eines Prozessors eine Anforderung mit Betriebsmittelzugriff verarbeitet wird, darf diese solange rechnen, wie es die Zeitscheibe zuläßt. Unabhängig davon, ob im konkurrierenden System ebenfalls eine Anforderung zur Bearbeitung ansteht, die das Betriebsmittel verwenden möchte, und die somit warten muß.

Innerhalb jeder Warteschlange gibt es, je nach Wahl der Zugriffswahrscheinlichkeiten p_1 oder p_2, zwei Sorten von Anforderungen, die mit und die ohne Betriebsmittelzugriff. Eine gleichzeitige Verarbeitung, und die damit verbundene unvermeidliche Blockierung von Prozessen, die beide das Betriebsmittel anfordern, soll in den Prozessoren beider Systeme vermieden werden. Dadurch wird ein gleichzeitiger Betriebsmittelzugriff ausgeschlossen. Da es sich hier um zwei Teilsysteme handelt, die unabhängig voneinander parallel ablaufen, benötigt man einen Mechanismus zur Sequentialisierung, um die beiden Teilsysteme nacheinander untersuchen zu können.

Dies soll unter anderem durch eine Prioritätssteuerung erfolgen, indem das System 1 immer vorrangig betrachtet wird. Das heißt, wenn an beiden Prozessoren gleichzeitig zwei Anforderungen anstehen, die um das Betriebsmittel werben, hat das System 1 Priorität. Das gilt auch prinzipiell, wenn zwei Ereignisse gleichzeitig stattfinden: wenn beispielsweise zum selben Zeitpunkt eine Anforderung in System 1 den Prozessor verlassen, und in System 2 eine Anforderung auf den Prozessor zugreifen möchte, wird im ersten Schritt das erste Teilsystem bearbeitet, bevor das zweite Teilsystem im zweiten Schritt betrachtet wird.

Durch die Prioritätsregelung wird vermieden, daß es zu Verklemmungsproblemen kommt. Man denke sich dabei nur folgende Situation: An Prozessor PR_2 steht eine Anforderung $A2_m$ mit Betriebsmittelzugriff an. Diese muß zunächst warten, da an Prozessor PR_1 zur selben Zeit ebenfalls eine Anforderung $A1_n$ mit einer Anmeldung für das Betriebsmittel ansteht. Ohne

Prioritätssteuerung die regelt, welche Aktion als nächstes ausgeführt werden soll, muß aber $A1_n$ ebenfalls warten, wegen $A2_m$! Daß heißt, man ist an einem „toten" Punkt angelangt, was unbedingt zu vermeiden ist.

2.3 Zielsetzung

An Hand des vorgestellten Systems sollen nun verschiedene Fragen diskutiert werden, auf die in den einzelnen Abschnitten des Kapitels „Simulationsergebnisse" noch näher eingegangen wird. Verwendete Fachbegriffe und Abkürzungen können vorab im „Anhang A" nachgeschlagen werden. Von Interesse sind insbesondere folgende Punkte:

- Wie verhalten sich die Wartezeiten und die Verweilzeiten der Anforderungen bei verschiedenen Parametereinstellungen?
- Wann befinden sich die beiden Teilsysteme im statistischen Gleichgewicht?
- Wie wirkt sich die Verkehrsdichte ρ und die Zugriffswahrscheinlichkeit p auf das Systemverhalten aus?
- Welchen Einfluß haben die Zeitscheibenlängen auf das Systemverhalten?
- Wie hängen die Warteschlangenlängen von den Systemeinstellungen ab? Wann und wie entsteht innerhalb einer Warteschlange ein Stau?
- Was passiert mit Anforderungen mit sehr großen Bedienungszeiten? Wann findet ein Aushungern der Anforderungen statt, und wie läßt sich das durch Verwendung größerer Zeitscheiben vermeiden?
- Welcher Zusammenhang besteht zwischen den Systemparametern und dem Durchsatz des Systems? Wann ist das System optimal ausgenutzt?
- Wie groß ist der Auslastungsgrad der Komponenten (Betriebsmittel und Prozessoren von System 1 und System 2)?

3 Implementierung

Die Implementierung des Programms erfolgte in PASCAL[6], auf einem IBM-kompatiblen PC des Rechenzentrums der Georg-Simon-Ohm Fachhochschule Nürnberg. Die Struktogramme der wesentlichen Teile des Algorithmus, sowie die Modulstruktur mit den darin enthaltenen Funktionen und die zugehörigen Quelltexte befinden sich im zweiten Band „Teil II der Diplomarbeit: Modulstruktur und Quelltexte des PASCAL Programms".

3.1 Sequentialisierung

Wie bereits in der Einleitung erwähnt wurde, geht es bei der Untersuchung um die Simulation zweier zeitunabhängiger Warteschlangensysteme, was man auch als statische Systeme bezeichnet, mit Hilfe der Monte Carlo Simulation. Der zentrale Punkt für eine Problemlösung ist, den zeitlichen Ablauf der Ereignisse der beiden, voneinander unabhängigen, parallel angeordneten Teilsysteme zu sequentialisieren. Das heißt, die zeitlichen Ereignisse beider Systeme müssen nacheinander verarbeitet werden, indem man die beiden Teilsysteme durch eine diskrete Ereignis-Simulation (discrete event simulation) modelliert.

Da es sich um zwei statische Systeme handelt, ist die Reaktion auf ein Ereignis nicht davon abhängig, wann es eintrifft, sondern davon, in welchem Zustand sich jedes Teilsystem, und seine Komponenten, zum Zeitpunkt des Eintreffens befindet. Beispielsweise ist der Prozessorzugriff einer anstehenden Anforderung in der Warteschlange davon abhängig, ob der Prozessor besetzt oder frei ist.

Der Zustand eines Systems ändert sich gegebenenfalls durch den Eintritt von Ereignissen an bestimmten, endlich vielen Zeitpunkten, beispielsweise den Zwischenankunftszeiten der Anforderungen. Die diskreten Zeitpunkte, an denen möglicherweise Zustandsänderungen auftreten, werden dabei über eine Art „Simulationsuhr" ermittelt, die ich von nun an als Timer bezeichnen möchte. Die diskrete Ereignis-Simulation, oder anders ausgedrückt, die Timer-Steuerung kann dabei auf zwei Arten erfolgen [Dom91]:

- Der Timer wird laufend um eine bestimmte (konstante) Zeiteinheit erhöht. Nach jeder Aktualisierung des Timers wird überprüft, welche Ereignisse während des Zeitintervalls eingetreten sind, die zu einer Veränderung des Systemzustands führen. Dieses Vorgehen bezeichnet man als periodenorientierte Zeitführung.
- Der Timer wird auf die Zeitpunkte gesetzt, an denen zukünftige Ereignisse eintreten. An Hand dieser Ereignisse wird der Einfluß auf das Systemverhalten untersucht. Dieses Verfahren bezeichnet man als ereignisorientierte Zeitführung.

[6] Anmerkung zur überarbeiteten Fassung: Das Programm liegt auch in Turbo C vor, was allerdings nicht Bestandteil der ursprünglichen Aufgabenstellung war.